修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院 情報理工学研究科　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	菅原 真	学籍番号	1631069
論 文 題 目	チェスの人工知能からなる multiple choice system の構築		

要　旨

本研究はチェスの人工知能(AI)の性能向上を目的として，チェス AI と畳込みニューラルネットワーク(CNN)を用いて multiple choice system の一種である double-Fritz with boss を構築した．Multiple choice system とは提示された複数ある意見の中から人間がより良いものを選択するようなゲーム AI に関する手法で，様々なボードゲームの AI の性能向上が報告されている．

Double-Fritz with boss の概要は次のようである．まず，チェス AI がその手番において最も有効と考える手(最善手)と，次に有効と考える手(次善手)を提示する．そして，チェス AI よりも弱い人間が一方を選択し，着手を行う．この一連の流れを繰り返して，ゲームを進行させる．

本研究では次のような 2 つの実験を行った．

1 つ目は，CNN により対局結果である勝ち，引き分け，負けの予測を行う実験である．もし，対局結果が正確に予測できるのであれば，期待勝ち点が高い手を選択し続けることで元のチェス AI と同等以上の性能を持たせることができるであろう．実験を行った結果，CNN にチェス AI が出力する評価値と駒の配置を入力することによって予測精度が向上することが分かった．またネットワークを大規模化することによっても有意に予測精度が向上することが確認された．

2 つ目は本研究で提案する，チェス AI と CNN から構成される double-Fritz with boss を用いた実験である．この手法を用いた AI は CNN によって最善手と次善手の期待勝ち点を算出し，より期待勝ち点が高い手を着手する．この AI と最善手のみを選択する AI で対局実験を行い，元のチェス AI に勝ち越すかを検証する．CNN を用いて構築した AI は最善手のみを選択する AI と比べて，期待勝ち点に有意な差が確認されなかった．また，提案した AI の性能は評価値をみてときおり次善手を選択するような方法とほぼ同等程度だった．その一方で，CNN の大規模化によって予測精度を大きく向上させることが出来れば，CNN を用いた AI の性能も向上させ得るという感触を得た．

電気通信大学 情報理工学研究科
情報・ネットワーク工学専攻 情報数理工学コース 修士論文

チェスの人工知能からなる multiple choice system の構築

平成 30 年 3 月 5 日

学籍番号 1631069

菅原 真

指導教員 保木邦仁

目 次

1　はじめに

人工知能 (Artificial Inteligence, AI) を作る際に, 達成すべき目標の 1 つとして「知的活動を伴うような問題を解決する人工物を生成すること」が挙げられる. 問題解決において, 人や AI が数ある行動の中からより状況に適したものを選択することは困難である. また, 一部の問題に対して人よりも優れた状況判断を行えるような AI はそう多くない.

そこで問題解決に取り組む人間の思考について着目する. 人間が問題に直面し, その解決に向けて行動する際, 経験や知識だけではなく他者の考えを参考にする. AI に関してもこれを参考にすることができるのではないかと考えた.

認知科学の分野などにおいて, 集合知という概念がしばしば用いられる. 集合知は, 「ある目的を達成するために共同で知的作業を行う個々の集団」と定義され [1], 多数の個体の知識の集積を利用して, 意思決定を行うことができる. このような集合知をゲームの AI を用いて構成した研究がいくつか存在し, その例として multiple choice system [2] や多数決法が挙げられる.

Multiple choice system はゲーム AI から提示された複数の意見に基づき人間が意思決定を行うような手法であり, その一種として double-Fritz with boss が提案されている. このシステムの概要は次のようである. まず, 意見を提示する AI をベース AI と呼ぶ. チェスのベース AI である Fritz が, 指し手の候補を 2 つボスに提示する. ここで提示された手の内ベース AI がもっとも有効と考える手を最善手, その次に有効と考える手を次善手とする. そして, その一方をボスが選択し着手する (図 1 参照). このシステムを用いて対戦実験を行ったところ, チェスの強さの指標とされる elo レーティングがベース AI と比較して 150 ほど上回ったとの報告がなされた. ここで, 意思決定を行うボスの elo レーティングはベース AI と比較して 450 ほど低い. しかし, この実験における elo レーティングは十分な対戦回数で測定されていないため統計的に有意な結果とは言えない.

その一方で, 多数決法は人間による意思決定が不要という長所を持ち, 千試合ほどの規模で, 有意に勝率を高くすることが確認されている. また複数種類のボードゲームにおいても, 多数決法の研究に一定の進展が見られる [3, 4, 5, 6, 7, 8, 9]. しかし, double-Fritz with boss のボスほどの高度な知的判断を伴うような候補の選択アルゴリズムは未だ提案されていない.

2　研究目標

本研究は, チェスのベース AI である Stockfish7 と AI によって構成されたボスを用いて double-Fritz with boss を構築し, ベース AI の性能向上を目指す. Stockfish7 は, オープンソースのソフトウェアの中で最も強いチェス AI の 1 つである. Stockfish7 は 2014 年 8 月に行われた, 世界でも上位のプロプレイヤーとの, ハンデキャップを背負った対戦で圧倒したことでも話題となった. 本来 double-Fritz with boss を構築するにあたって, ベース AI には Fritz を用いることが望ましい. しかし, Fritz は商用プログラムでソースコードが公開されていないため, Stockfish7 をベース AI とした. また, ボスである AI はニューラルネットワーク (Neural Network, NN) から構成する. 本研究で取り組んだ具体的な目標は以下のようである.

- NN に, Stockfish7 の自己対戦の結果である勝ち・引き分け・負けを予測させることを試みる. また NN の大規模化に伴う勝ち, 引き分け, 負けの予測精度の変化の様子を検証する.

- NN を用いてボスを構築し, よりボスが勝ちやすいと判断する手を実際に着手することで, ベース AI の性能が向上するかを調査する. NN を用いて構築した AI とベース AI との対戦を行い, ベース AI に勝ち越すことが出来るかを検証する. また対戦から得られたデータから NN を用いたボスの性能を定量的に分析する.

3　関連研究

3 章では本研究に関する先行研究についての調査結果について述べる．本項で述べる先行研究は本研究で multiple choice system を構築するにあたって参考にしたものである．

3.1　multiple choice system

Althöfer は, ゲーム AI と人間による集団の知恵を応用したシステムである multiple choice system を提案した [2]．Althöfer は double-Fritz with boss (図 1 参照), 3-hirn (図 2 参照) や list-3-hirns (図 3 参照) などの様々な形態の multiple choice system を用いて, チェスや碁の対戦実験を行った．

ここで各形態の multiple choice system について説明する．Double-Fritz with boss は MultiPV によってベース AI である Fritz に最善手と, 次善手を生成させ, 一方をボスである人間が選択する．3-hirn はそれぞれ異なる 2 つのベース AI が提示した 2 つの候補手のうち一方をボスである人間が選択する．List-3-hirns は 3-hirn と double-Fritz with boss を組み合わせたシステムである．2 つのベース AI を用意し, それらの AI に double-Fritz with boss と同様に最善手と次善手を提示させる．2 つの AI から提示された計 4 つの候補手の中からボスである人間が指し手を選択する．Althöfer によって行われたそのほとんどの実験において, 統計的に有意な結果ではないが, 基となるベース AI よりも概ね強さが向上したことが確認された．本研究はこの研究の double-Fritz with boss から着想を得て始めたものであるが, 先行研究では意思決定を行う主体であるボスが人間である一方, 本研究では NN を用いて意思決定を行っている点に相違が見られる．

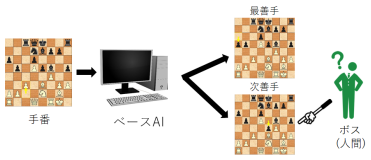

図 1: Double-Fritz with boss

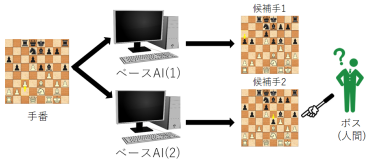

図 2: 3-hirn

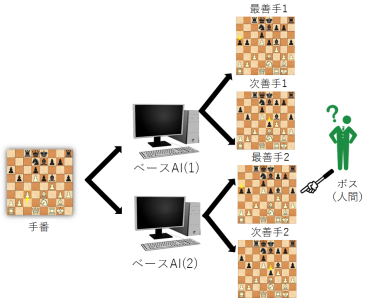

図 3: List-3-hirns

multiple choice system の例

3.2　チェス AI を題材とした集合知

ゲーム AI 研究の題材とされているチェスは世界各国で最も親しまれているゲームの 1 つである．欧州では計算機の発展とともに, チェス AI に関しての研究が盛んに行われてきた．現在はプロプレイヤと同等の実力を持つチェス AI が多数存在する．様々な手法におけるチェス AI 開発の試みとともに, チェス AI を題材とした集合知に関する研究も進められてきた．

Spoerer らは異種合議法がどのような条件下で有効に働くかをチェスで調査した [3]．異種合議法とは, 異なる複数のベース AI が個々の思考によって導いた候補手に投票し, 指し手を多数決によって決定する手法である．これは Althöfer が提唱した手法, 3-hirn をベースに機械的に構

築したシステムである. 以後, 記述する様々な合議法はそのほとんどが 3-hirn から着想を得たものである. この研究の合議システムはチェス AI である, Stockfish7, Toga, Bobcat の 3 つのベース AI によって構築されている. 合議システムは, 基本的に最も投票数の多い手を選択し, 意見が三つに割れたとき最も強いベース AI であるリーダー AI, Stockfish7 の手を選択する. 各ベース AI は指し手の探索の深さに制限が課されていて, 探索の深さの制限を変えることによって合議システムを構成するベース AI の強さを変え, これに伴い合議システムの強さがどう変化するかを調査している. 実験によって, 合議システムの強さがリーダー AI の強さを上回る条件が, ベース AI の強さをほとんど等しくすることであることや, ベース AI が強いほど意見の不一致が多発し, リーダー AI の意見が却下される頻度が多くなることが確認された. この先行研究と本研究の類似点は, チェス AI である Stockfish7 をベース AI として用いている点である. 先行研究では意思決定に, 投票数やリーダの投票などの単純な手法を用いているが, 本研究では機械学習を用いている点に相違がみられる.

保木らは乱数合議法やその一種である楽観合議法に着目し, どのような条件下でこれらの合議法がより有効に働くのかをチェスで調査した [4, 5]. 乱数合議法は, 次のようなものである. まずベース AI が算出する評価値に乱数を加えるベース AI を複数用意する. そして複数のベース AI はそれぞれ異なるシード値によって乱数の分布を決定する. 各ベース AI に最も評価値の高い手に投票させ, その中で最も投票数の多い手を選択する. 投票数が最大である手が複数あるような場合は, その中で最初に投票された手が選択される. また, 楽観合議法とは基本的に最も評価値の高い手を選択し, 最も高い評価値を持つ候補手が複数存在するとき, それらの中からランダムに指し手を選択する手法である. 実験は, 評価値に加える疑似乱数の分散やベース AI が行う候補手の探索の節点数, 合議に用いるベース AI の数を変えることで, 2180 問のチェスプロブレムの次の一手の予測の正答率がどのくらい変わるのかを調査した. その結果, 疑似乱数の分散が大きいほど楽観合議法の性能は向上し, 乱数合議法の性能は低下する傾向が見られた. 意見を選択するにあたって評価値を基準としている点に本研究との類似が見られる. また, 本研究と先行研究との相違点を次の観点から 2 つ述べる. 1 つめは先行研究の意見の選択方法が, 評価値の大小関係や票の数のみに基づく, 比較的単純なものである点である. 2 つめは, チェス AI の性能の向上を調査するために本研究では対戦実験を行っているが, 先行研究ではチェスプロブレムの次の一手の予測と正答の一致率を測っている点である.

3.3 チェス以外のボードゲームの AI を題材とした集合知

チェス以外のボードゲームの AI を題材とした集合知に関する研究も多数存在する. その一つに 5 五将棋のベース AI を用いた研究がある. 5 五将棋とは, 5 行 5 列のマス目を持つ盤と六種類の将棋の駒を用いて行われるゲームである. 小幡らは 5 五将棋と将棋の AI を題材として, AI が出力する複数の異なる意見から一つの意見を多数決により導く方法を研究した [6]. 5 五将棋のベース AI には千分ノ壱里眼が, 将棋のベース AI には Bonanza が用いられた. 複数の異なる意見は, 静止探索の使用の有無やそれぞれのベース AI が持つ評価関数の駒の価値を変更することにより生成している. 5 五将棋での実験は, 多数決法を用いた AI の勝率が単体のベース AI を用いたときの勝率と比べて有意に上回る結果となった. また, 多数決法において強いものをリーダー AI にする方がより強くなる可能性があることを示唆した. これらの実験結果はゲーム AI の並列化手法として, 多数決法が有効であることを示すものであった.

小幡らの研究を皮切りに, 将棋 AI 研究のアプローチの 1 つとして将棋 AI を用いた multiple choice system の研究が盛んに行われるようになった.

伊藤らは複数の独立した思考システムが個々に導いた候補手の中から一つの手を選択する手

法として, 異種合議法や乱数合議法の提案と評価を行った [7]. 実験の結果, どちらの合議システムも合議システムを構成するベース AI と比べて有意に性能が向上した. また, 異なるベース AI に対しても, 疑似乱数を用いた合議システムが高いパフォーマンスを発揮できることが示され, 異種合議法と乱数合議法の 2 つの合議手法が, ベース AI の性能向上という観点において有効であることを示した. 本研究と比較して先行研究では, 各ベース AI の最も評価値の高い手から指し手を選択している点に類似が見られる. また本研究では意思決定法に機械学習を用いているが, 先行研究では多数決法といった単純な手法を用いている点に相違が見られる.

杉山らは疑似乱数を用いる乱数合議法の 1 つとして, 楽観合議法を提案した [8]. 複数の異なる意見は, それぞれのベース AI が持つ評価関数に小さな乱数を加えることによって生成している. 実験の結果, Bonanza, GPS 将棋ともに, 合議システムの勝率が, ベース AI と比較して有意に上回った. 本研究では評価値を参考にして意思決定を行っているが, 先行研究では意見を選択する基準として評価値を用いている点に類似がみられる. また本研究では手番情報と評価値を基に一つの意見を選択しているが, 先行研究では評価値のみから意思決定が行われている点に相違がみられる.

将棋やチェス以外の思考ゲームの AI を題材とした研究も行われている. Danilo らはチェッカーの AI を用いた乱数合議法の有効性を調査した [9]. 合議システムはチェッカーのベース AI である Samuel によって構築されている. Samuel は探索の深さ制限によって性能を変えることができ, 探索の深さ制限, ベース AI の数や評価値に加える乱数の分布などを変化させることで, 合議システムの性能の変化を調査した. 実験では, 合議システムによって指し手を決定する AI と, 乱数値を加えず探索の深さを合議システムを構成するベース AI と同じように設定した元の Samuel を対戦させることで勝率を計測した. その結果, 用意した複数のベース AI がすべて弱いような場合は合議システムの性能が向上し, 強いような場合は性能が低下することが確認された. またベース AI の数を変えることによる性能の向上は確認されなかったが, 相手の性能がはっきりしているような場合において, 乱数の分布を調整することで性能を向上させうることが示された.

3.4 NN を用いたボードゲームの AI の実現

近年ゲームにおいて NN が強い AI の開発に貢献している. AlphaZero や DeepStack, TD-gammon などは, 状態価値関数, 行動価値関数を精密に近似しており, プロよりも強い実力を持っている [10, 11, 12].

例えば, AlphaZero は, チェスにおける成功例の 1 つである [10]. AlphaZero は現在の手番を含む 8 手前までの駒の配置や, 手番, 手数, キャスリングの有無, 50 手ルールのカウントなどを畳み込み NN(Convolutional Neural Network, CNN) に入力して, 着手確率や期待勝ち点を推定する. この研究で用いられた CNN はフィルタ数が 256, 層の数が 20 以上となっている. 本研究で用いた CNN の規模は, AlphaZero のものより小規模なものである.

4　基礎知識

4 章では実験に用いたプログラムの説明と, 本実験で用いた最適化手法について述べる.

4.1　チェス AI Stockfish7

ベース AI に用いたオープンソースのソフトウェアである Stockfish7 について説明する. Stockfish7 は以下のような特徴を持つ.

1. UCI プロトコル対応

   対戦中, チェス AI はゲームに関する情報を保持するサーバと通信を行う. サーバはチェス AI に対して, 現手番までに指された手の系列を返し, チェス AI はサーバに対して指し手を表す文字列を返す. このやりとりを繰り返すことで対戦が行われる. これらのやり取りを統一するために, 作られた通信プロトコルが Universal Chess Interface (UCI) プロトコルである.

2. α-β pruning による後ろ向き枝刈りと前向き枝刈り

   Stockfish7 は探索時に各種枝刈りを行っている. 枝刈りには, ヒューリスティックに基づいて行う前向き枝刈りと探索時の情報を活用する後ろ向き枝刈りがある. α-β pruning は後ろ向き枝刈り法の一種である. ミニマックス木の現節点の評価値を求めるには末端節点の評価値が必要となるが, 全ての節点を探索する必要はない. ミニマックス木はある節点の評価値を求めるとき, 後続節点の評価値の最大値, もしくは最小値のみを必要とする. そのため, 評価値の上限値と下限値の範囲がわかっていればよい. そこで探索量を減らすために取り入れられた評価値の上限値を α 値, 下限値を β 値, α値とβ値 の区間を探索窓と呼ぶ. 探索時に, 探索窓から外れた手の枝刈りを行うことで, 探索量が削減される. また, さらなる探索の効率化を図るために aspiration windows という α-β pruning における探索空間を減らす手法が取り入れられている. この手法によって探索窓の範囲を微少に狭め, 枝刈りを効率よく行う. また Stockfish7 には, 最善手の探索を終えて持ち時間に余裕があり, 次項で述べる MultiPV が初期値の 1 以外の自然数に設定されている場合において,β 値を α値とポーン2 駒分の価値の差とすることによって探索窓の区間を狭めて探索を行う手法である null window search が実装されている. Stockfish7 は探索の効率化のため他に, ある深さまでに最善手の評価値を追い抜けなさそうな手を枝刈りする futility pruning, 駒の取り合いが終わるまで探索を続行する静止探索なども組み込まれている.

3. MultiPV 機能対応

   チェス AI, Stockfish7 は MultiPV 機能が実装されている. 通常 α-β pruning を用いる際, 最善応手系列の末端節点の評価値しか知ることは出来ないが, MultiPV 機能を用いることで最善手の他に, 次善手の最善応手系列と末端節点の評価値を求めることが出来る. しかし MultiPV 機能を用いることで, 探索時間が通常より多くかかるという欠点を持っている.

Stockfish7 は探索開始からサーバに指し手の文字列を返すまで, 探索途中の最善応手系列とその末端節点の評価値をログとして出力する. 本実験では MultiPV 機能を用いた探索を行わせているため, Stockfish7 は交互に最善手の最善応手系列とその末端節点の評価値, 次善手の最善応手系列とその末端節点の評価値を出力する. Stockfish7 により出力されたある節点における最善手, 次善手に関するログと指し手の文字列を返すまでの流れを例として挙げる.

```
info depth 11 seldepth 18 multipv 1 score cp 34 nodes 162334 nps 1531452 tbhits 0 time 106 pv e2e4 e7e6
b1c3 d7d5 e4d5 e6d5 d1e2 c8e6 d2d4 b8c6 g1f3 g8f6

Info depth 11 seldepth 18 multipv 2 score cp 25 nodes 162334 nps 1531452 tbhits 0 time 106 pv g1f3 d7d5
e2e3 g8f6 d2d4 e7e6 f1e2 b8c6 e1g1 f8e7 c1d2 e8g8 b1c3

info nodes 204568 time 134

bestmove e2e4
```

図 4: Stockfish7 によって出力されたログの例

図 4 は初期節点から探索節点数を 200,000 に制限して探索を行った時の Stockfish7 の出力の一部である．上から一番目の文字列が最善手に関するログ, 二番目の文字列が次善手に関するログ, 三番目の文字列が探索ノード数と探索時間を表すログ, 最後の文字列が指し手を表す文字列となっている. 候補手に関する文字列は現時点での探索深さ, 最善応手系列やその末端節点における評価値などが出力されている．基本的にはログで出力されている multipv が 1 となっている最善手が指し手として選択される. しかし最善手, 次善手以外が指し手として選択される場合が存在する. Stockfish7 が aspiration windows や null window search を取り入れているためである．これらの手法を用いることで探索空間の削減が行える一方, まれに探索している節点の評価値が α 値, β 値の範囲から外れてしまうことが起こる. このような現象を fail-high , もしくは fail-low と呼ぶ. Stockfish7 は fail-high, もしくは fail-low が起こった場合, 再探索を行うよう実装されている. そのため fail-high, fail-low が起こり, サーバに返される最善手がログに出力されている最善手と異なる場合がある. しかし本研究では, 候補手を最善手と次善手に限定したいため, 出力ログから得られる最善手, 次善手の 2 つから指し手の選択を行う.

4.2 Opening-book について

チェス AI 同士を複数回,　対戦させる場合, 初手から対戦が決着するまでチェス AI に着手させるとチェス AI がランダム性を多分に含まない限り, 同じようなゲームが繰り返されることとなる. このようにゲーム内容の重複が起こると勝率に偏りが生まれ, 正確にチェス AI の性能を計測することが出来ない. そこでゲーム内容の重複を避けるために, 序盤の数手を上級者のプレイヤの棋譜から構築されたデータベースを用いて指すことが多い．このようなデータベースを opening-book と呼ぶ. いくつかのチェス AI は opening-book があらかじめ組み込まれているが, 本実験ではゲーム内容の重複を避けるために, Polyglot[1]と呼ばれるフリーソフトウェアを用いた. Polyglot は, サーバから送られてくる情報を UCI プロコトルの形式に変換することが出来る他, 複数の既存の棋譜から opening-book を構築する機能が実装されている. 本実験では, サーバとベース AI との通信に利用するほかに, インターネットサイト PGN Mentor[2]から得られた上級者プレイヤの棋譜およそ 27 万局から opening-book を構築した. Opening-book の構築に用いる指し手は, 本実験では全ての棋譜のうち 12 局以上で指されている手とした.

[1]http://wbec-ridderkerk.nl/html/details1/PolyGlot.html
[2]https://www.pgnmentor.com/files.html

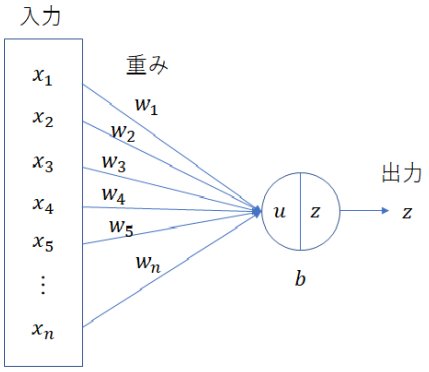

図 5: ニューロンユニット

4.3 NN

脳の神経細胞のことをニューロンと呼ぶ. ニューロンは他のニューロンから電気信号が伝達される. 入力は各々異なる伝達効率を持ち, ニューロンは各信号の電位と伝達効率の値との積和を受け取る. この伝達効率のことを重みと呼ぶ. ニューロンの電位が閾値を超えた際, ニューロンは他のニューロンに対して電気信号を伝達する. この現象をニューロンが発火したと呼ぶ. ニューロンを数式的にモデル化したものをニューロンユニットと呼び, 図 5 のように表すことができる. このユニットが受け取る総入力 u は各入力 $x_1, ..., x_n$ に対応した重み $w_1, ..., w_n$ やバイアスとよばれる値 b をもとに式 1 のように計算される.

$$u = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

このユニットの出力 z は活性化関数 f によって, 式 2 のように計算される.

$$z = f(u) \tag{2}$$

本研究では活性化関数は, シグモイド関数や線形正規化関数 (Rectified Linear Unit, ReLU) を用いた. シグモイド関数は式 3, ReLU 関数は式 4 で表される.

$$f_{\mathrm{sig}}(u) = \frac{1}{1 + \exp(-u)} \tag{3}$$

$$f_{\mathrm{ReLU}}(u) = \begin{cases} 0 & (n < 0) \\ u & (\text{otherwise}) \end{cases} \tag{4}$$

これらの活性化関数を用いることで式 1 のような線形関数の出力を非線形関数の出力に変換することが出来る.

4.3.1 順伝播型 NN

4.3.1〜4.3.6 の内容は講談社から出版されている「機械学習プロフェッショナルシリーズ 深層学習」 [13] の内容から本実験に用いた手法についての記述を要約したものである. 4.3.7 の内容は「数式で書き下す Convolutional Neural Networks (CNN) 」[1]の内容を要約をしたものである.

[1]http://blog.yusugomori.com/post/129688163130/

順伝播型 NN は多数のニューロンユニットから成る層から構成され, 隣接した層間のみに結合をもつ NN である. この NN は入力側の層から出力側の層に伝播する. ここで順伝播型 NN の $n+1$ 層における, あるニューロンユニットの出力を示す. n 層の i 番目のニューロンユニットの出力を x_i, $n+1$ 層の j 番目のニューロンユニットの出力を y_j, バイアスを b_j, n 層 i 番目のニューロンユニットから $n+1$ 層 j 番目のニューロンユニットに対する重みを w_{ij}, ある入力 p を与えたときの活性化関数を $f(p)$ としたとき, 出力 y_j は次式で示される.

$$y_j = f\left(\sum_i x_i w_{ij} + b_j\right) \tag{5}$$

4.3.2 勾配降下法

NN の出力値は入力と重みやバイアスなどのパラメータによって決定される. NN はこれらのパラメータを適切に調整することによって連続的な関数を近似することができる. ここで多クラス分類を行う NN について考える. 入力 X_i とこれに対応する K 個のラベル $\boldsymbol{d}_i = (d_{i1}, ..., d_{iK})(K \geq 2)$ の組から構成される学習データ集合 $T = \{(X_1, \boldsymbol{d}_1), ..., (X_N, \boldsymbol{d}_N)\}$ を用いて NN の出力値 $\boldsymbol{z} = (\boldsymbol{z}_1, ..., \boldsymbol{z}_N)$ に従属するような目的関数を最小化する. ここで, ラベル $\boldsymbol{d}_i$ の各成分は各クラスに対応し, クラスが真であったときのみ 1, それ以外は 0 を示すものとする. 入力を伝播して得られた出力値 $\boldsymbol{z}_i = (z_{i1}, z_{i2}, .., z_{iK})$ とラベル $\boldsymbol{d}_i$ から目的関数となるような誤差関数 E_i を求め, パラメータの調整を行う. 本研究では誤差関数は式 6 で示す交差エントロピー誤差を用いた.

$$E_i = -\sum_{k=1}^{K} d_{ik} \log z_{ik} \tag{6}$$

$$E = \sum_{n=1}^{N} E_n \tag{7}$$

式 6 で示した E_i はデータ点 1 つの交差エントロピー誤差, E はデータ全体の交差エントロピー誤差を表している. 重みの変更量を Δw_{jk} とする. 重みの変更量は,

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \tag{8}$$

で示される. η は学習率である. 各パラメータを出力層から更新していくことで, 局所最適解に近づけることができる. この方法を勾配降下法と呼ぶ. また, このようなパラメータの調整を学習と呼ぶこととする. 活性化関数を通す前の $n+1$ 層 j 番目のニューロンユニットの出力値 u_j は次式で示される.

$$u_j = \sum_i x_i w_{ij} + b_j \tag{9}$$

合成関数の微分から, 重みの変更量は次式のように変形できる.

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial u_j} \frac{\partial u_j}{\partial w_{jk}}$$

$n+1$ 層の重みを $\boldsymbol{w} = (w_{00}, ..., w_{0B}, w_{10}, ..., w_{AB})$ としたとき, $n+1$ 層の重みの変更量は $\Delta \boldsymbol{w} = (\Delta w_{00}, ..., \Delta w_{0B}, \Delta w_{10}, ..., \Delta w_{AB})$ で表され, これを $n+1$ 層の重みの勾配と呼ぶ. 現在の重みを $\boldsymbol{w}_t$, 更新後の重みを $\boldsymbol{w}_{t+1}$, 現在の重みの勾配を $\Delta \boldsymbol{w}_t$ としたとき, 下式のように重みを更新する.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \Delta \boldsymbol{w}_t \tag{10}$$

重みの初期値 $\boldsymbol{w}_0$を適切に定め, $\boldsymbol{w}_1, \boldsymbol{w}_2, ...$ と更新を繰り返すことで誤差関数 E が最小となるような重みを得ることができる. バイアス b_j の勾配も重みと同様に求まる.

パラメータの更新に用いるパラメータの変更量は全てのデータ集合についてではなく, ミニバッチと呼ばれる一部のデータ集合についても求めることができる. このように一部のデータ集合によるパラメータの調整を確率的勾配降下法 (Stochastic Gradient Descent, SGD) と呼ぶ.

4.3.3 慣性項について

重みやバイアスなどのパラメータの更新の際, 前回のパラメータの変更量を用いることで収束を早め, 振動を抑える手法があるその変更量にあたる項を慣性項と呼ぶ. 慣性項の係数 μ は 1.0 以下の正の実数を用いる. 慣性項を取り入れた重みの更新式は次式のように示される.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \Delta \boldsymbol{w}_t + \mu \Delta \boldsymbol{w}_{t-1} \tag{11}$$

本実験で SGD を用いる際は, 学習率 $\eta = 10^{-5}$, $\mu = 0.9$ とした.

4.3.4 Adaptive moment estimation (Adam) について

重みの更新の度合いを定める学習率を適切に定めることは難しい. 例えば学習率が大きすぎると収束を妨げ, 小さすぎると収束が遅くなってしまう. そこでパラメータに対して適切に学習率を定める手法が近年盛んに研究されていてその 1 つに Adam という手法 [14] がある. 本研究の実験では基本的に Adam を用いた. Adam による重みパラメータの更新を例として以下に示す. Adam は次のような手順でパラメータを更新する. 以下に示す Adam のパラメータの更新手順は元論文 [14] の内容を一部要約したものである. Adam は学習率の他にパラメータ β_1, β_2, ϵ を用いる. 本実験では元論文で述べられている値 $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ を用いた. 一次モーメントベクトル $\boldsymbol{m}_0$, 二次モーメントベクトル $\boldsymbol{v}_0$, タイムステップ t を $\boldsymbol{m}_0 = \boldsymbol{0}, \boldsymbol{v}_0 = \boldsymbol{0}, t = 0$ とする.

1. まず, タイムステップ t の数値を $t + 1$ にする.
2. 次に, タイムステップ t における重みの勾配 $\Delta \boldsymbol{w}_t$ を求める.
3. 一次モーメントの概算値を次式のように更新する.

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1) \Delta \boldsymbol{w}_t \tag{12}$$

4. 二次モーメントの概算値を次式のように更新する. 次式の $\Delta \boldsymbol{w}_t^2$ は勾配 $\Delta \boldsymbol{w}_t$ の各成分を二乗した数値をもつベクトルである.

$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2) \Delta \boldsymbol{w}_t^2 \tag{13}$$

5. 一次モーメントのバイアス補正を行った推定値を次式のように求める.

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \tag{14}$$

6. 二次モーメントのバイアス補正を行った推定値を次式のように求める.

$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \tag{15}$$

7. タイムステップ t における重みパラメータを次式のように更新する.

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} \tag{16}$$

8. 1 に戻る.

このような手順で各パラメータを更新する. 本研究では学習率 η を 10^{-5} とした.

4.3.5 CNN

CNN は主に画像認識に用いられる順伝播型 NN である.3 章で述べたように, 近年 Silver らが開発した CNN と MCTS を用いたチェス AI, AlphaZero がコンピュータプレイヤの中でも当時最も強いとされていた Stockfish8 を打ち破ったことで話題となった. 今までに述べた NN は各ニューロンユニットが隣接層のユニット全てと結合している, 全結合層によって構成されていた. 一方 CNN は, 隣接間のニューロンユニットと部分的に結合を持つ畳み込み層と全結合層などが組み合わさって構築された NN である. CNN は人間の視覚野における神経細胞と同様に, 特定の場所に特定のパターンが入力されたときニューロンユニットが発火する. 各畳み込み層は単数, もしくは複数のフィルタと呼ばれる $H_{\mathrm{f}} \times W_{\mathrm{f}}$ の実数値から成るベクトルを持ち, 各ユニットに対してではなく各フィルタが重み, バイアスのパラメータを持つ. 畳み込み層は $H_{\mathrm{g}} \times W_{\mathrm{g}}$ のニューロンユニットによって構築される. 本論文では各ニューロンユニットを画素, ある層における全てのニューロンユニットの集まりを画像と呼ぶ.

n 番目の畳み込み層の i 行 j 列 目の k チャネルの画素を $x_{i,j,k}$, m 番目のフィルタを通して得られた $n+1$ 番目の畳み込み層の i 行 j 列 目の画素を $y_{i,j,m}$, m 番目のフィルタが持つバイアスを b_m, p 行 q 列 目の各フィルタ $(k = 0, ..., K-1)$ の重みを $w_{p,q,k,m}$ としたとき, 畳込み層における出力 $y_{i,j,m}$ は次式で示される.

$$y_{i,j,m} = \sum_{k=0}^{K-1} \sum_{p=0}^{H_{\mathrm{f}}-1} \sum_{q=0}^{W_{\mathrm{f}}-1} (x_{i+p,j+q,k} w_{p,q,k,m}) + b_m \tag{17}$$

式 17 のように畳み込み層では画像にフィルタを重ね, 重なり合う画素とフィルタの重みの積を求めフィルタ全体で和を求める. 従ってフィルタがはみ出ないよう計算が行われるが, 出力先の画像のサイズは $\left(H_{\mathrm{g}} - \lfloor \frac{H_{\mathrm{f}}}{2} \rfloor \times W_{\mathrm{g}} - \lfloor \frac{W_{\mathrm{f}}}{2} \rfloor\right)$ となり, 入力された画像より小さくなる. そのため畳み込み層による計算が行われるたび, 出力先の画像のサイズが小さくなり, 画像のサイズによって CNN の層の最大数が限定されてしまう. これを防ぐため, 入力画像の外側であるふちに一定の値を持った画素を付け加えることで, 出力先の画像のサイズが元の入力画像のサイズと同じになるようにする, パディングと呼ばれる手法が存在する. また式 17 は 1 画素ずつフィルタをずらしながら積和を計算した式となっているが, 畳み込みの計算は数画素ずつずらしても計算することができる. ストライドは畳み込みの計算の適用間隔を表している. ストライドの値を s としたとき, ストライドを適用した畳込み層の計算式は次式のように示される.

$$y_{i,j,m} = \sum_{k=0}^{K-1} \sum_{p=0}^{H_{\mathrm{f}}-1} \sum_{q=0}^{W_{\mathrm{f}}-1} (x_{si+p,sj+q,k} w_{p,q,k,m}) + b_m \tag{18}$$

ストライドを適用した場合, 出力先の画像サイズは約 $1/s$ となる.

4.3.6 全結合層における逆誤差伝播法

ここでは n 層における全結合層の逆誤差伝播について考える. 全結合層の重みの変更量の式は n 層目の重み w_{ij}^n と出力 z_j^n を用いて次式で示される.

$$\frac{\partial E}{\partial w_{ij}^n} = \frac{\partial E}{\partial z_j^n}\frac{\partial z_j^n}{\partial w_{ij}} \tag{19}$$

n 層目の出力 z_j^n の変化は $n+1$ 層目の総出力 $z_k^{(n+1)}$ に影響をもたらす. z_i^n が誤差 E に与える影響も総出力 $z_k^{(n+1)}$ を用いて表すことができるため, 式の右辺の第一項は微分連鎖により次式のように変形できる.

$$\frac{\partial E}{\partial z_j^n} = \sum_k \frac{\partial E}{\partial z_k^{(n+1)}}\frac{\partial z_k^{(n+1)}}{\partial z_j^n} \tag{20}$$

誤差 E に対する n 層目の出力 z_i^n の微分 $\frac{\partial E}{\partial z_j^n}$ を δ_i^n と表すと, 式 21 は次式のように変形できる.

$$\delta_i^n = \sum_k \delta_k^{(n+1)}\frac{\partial z_k^{(n+1)}}{\partial z_j^n} \tag{21}$$

ここで $z_k^{(n+1)} = \sum_j w_{jk} z_j^n = \sum_j w_{jk} f(u_j^n)$ であることに注目すると, $n+1$ 層目の出力 $z_j^{(n+1)}$ に対する n 層目の出力 z_j^n の微分 $\partial z_k^{(n+1)}/\partial z_j^n$ は次式のようになる.

$$\frac{\partial z_k^{(n+1)}}{\partial z_j^n} = w_{jk} f'(u_j^n) \tag{22}$$

$f'(u_k)$ は活性化関数 $f(u_j)$ に対する u_j の一次微分を表している. 式 22 を利用することで式 21 は次式のように変形できる.

$$\delta_i^n = \sum_k \delta_k^{(n+1)}(w_{jk} f'(u_j^n)) \tag{23}$$

式の右辺の第二項は $f'(u_j) = z_i^{(n-1)}$ となるため δ_i^n を式 4.3.6 に代入することで勾配は次のように求まる.

$$\frac{\partial E}{\partial w_{ij}^n} = \delta_j^n z_i^{(n-1)} \tag{24}$$

出力層の δ_j^N は $\delta_j^N = \partial E/\partial u_j$ で求まる.

4.3.7 畳込み層における逆誤差伝播法

畳み込み層の重みの変更量 $\Delta w_{p,q,k,m} = \partial E/\partial w_{p,q,k,m}$ を求める. 畳込み層における重みの変更量は微分連鎖によって, 次式のように示される.

$$\Delta w_{p,q,k,m} = -\eta\frac{\partial E}{\partial w_{p,q,k,m}} \tag{25}$$

$$= -\eta \sum_{i=0}^{H_g-H_f-1}\sum_{j=0}^{W_g-W_f-1}\frac{\partial E}{\partial y_{i,j,m}}\frac{\partial y_{i,j,m}}{\partial w_{p,q,k,m}} \tag{26}$$

$$= -\eta \sum_{i=1}^{H_g-H_f}\sum_{j=0}^{W_g-W_f-1}\frac{\partial E}{\partial y_{i,j,m}} x_{i+p,j+q,m} \tag{27}$$

ここで $\delta_{i,j,m} = \partial E/\partial y_{i,j,m}$ について考える. 畳込み層に続く次の層が全結合層であった場合, $\delta_{i,j,m}$ は式 23 を用いることで求まる. 次に畳込み層に続く次の層が畳込み層である場合, 前の畳込み層からの出力 $x^n_{i,j,m}$ に関する変更量 $\delta^n_{i,j,m} = \partial E/\partial x^n_{i,j,m}$ を考える必要がある. $x^n_{i,j,m}$ がフィルターを通して次の層に出力されることを考慮し, 次の層の畳込み層の出力に関する誤差を $\delta^{(n+1)}_{i,j,m}$ とすると, $\delta^n_{i,j,m}$ は次式のように示される.

$$\delta^n_{i,j,m} = \sum_{k=0}^{K-1} \sum_{p=0}^{H_\mathrm{f}-1} \sum_{q=0}^{W_\mathrm{f}-1} \frac{\partial E}{\partial y_{i-p,j-q,m}} \frac{\partial y_{i-p,j-q,m}}{\partial x_{i,j,m}} \tag{28}$$

$$= \sum_{k=0}^{K-1} \sum_{p=0}^{H_\mathrm{f}-1} \sum_{q=0}^{W_\mathrm{f}-1} \frac{\partial E}{\partial y_{i-p,j-q,m}} w_{p,q,k,m} \tag{29}$$

$$= \sum_{k=0}^{K-1} \sum_{p=0}^{H_\mathrm{f}-1} \sum_{q=0}^{W_\mathrm{f}-1} \delta^{(n+1)}_{i-p,j-q,m} w_{p,q,k,m} \tag{30}$$

5 CNNを用いたStockfish7による自己対戦の勝ち, 引き分け, 負けの予測

この章では, 構造の異なる複数のCNNによってベースAIを用いたゲームの結果である勝ち, 引き分け, 負けの予測を行う.

5.1 実験方法

この節では, CNNに入力するデータ集合の採取や, 手番の数値化, 構築したCNNの構成などについて説明する.

5.1.1 学習データ集合とテストデータ集合について

チェスの状態数はおよそ10^{47}ほど存在する[2]. 本研究で構築するCNNは, これら全ての状態に対して, 適当な予測を返すものであることが望ましい. しかし, これらの状態全てを採取するのは現実的でない. そのため, 実際のゲームで到達しそうな手番を重点的に用いて学習を行いたい. そこで本研究では実際のゲームで到達しそうな手番を採取するため, opening-bookとベースAIであるStockfish7の自己対戦を利用する.

CNNの学習に用いる学習データ集合は入力X_iとラベルd_iの組を要素とする集合である. 入力X_iはチェスの手番m_iに対応しており, ラベルd_iはその手番の後, ベースAIによる自己対戦を行った結果から得られる勝ち, 引き分け, 負けである. 手番m_iから入力X_iを作る方法については次項で説明する.

学習データ集合の採取は次のように行う. 通常の対戦で用いられる初期配置から白を先手, 黒を後手として対戦を開始する. まず手番が, opening-bookに登録されている場合, これから指し手を選択する. 登録されていなければ, 98%の確率でベースAIの最善手, 2%の確率で次善手を選択する.

ここで対戦開始から終了までの手番とその対戦結果を$m_1 \cdots m_{T-1} r$と表し, 最後にopening-bookから着手した手番をm_b, 最後に次善手を選択した手番をm_sとする. 但し, 1つの対戦で次善手が選択されていなければ, $s = 0$とする. ここで最善手のみを選択して対戦結果が得られた手番を採取したい. もし, $b \geq s$ならば手番$m_{b+1} \cdots m_{T-1}$と結果rを採取する. $b < s$ならば, 最後に次善手が選択された手番から終局までの偶数手番$m_s m_{s+2} \cdots$と結果rを採取する. 次善手が一度も選択されない対戦は2割以下であった.

ベースAIの1手番ごとの探索節点数は20万節点に制限した. また, 区間[-1000, 1000]に属さない手番は採取しないものとした. さらに, 最善手を選択した手番と次善手を選択した手番の比率がおおよそ1となるように, 最善手を選択した手番は一定の確率で学習時に棄却して, CNNの学習を行った.

テストデータ集合も学習データ集合と同様に生成する. 違いは1つの対戦から採取可能なデータの組のうち, ランダムに1組のみを採取した点にある. テストデータ集合をこのように採取することで, CNNの汎化性能を調査する.

今回の実験では学習データ集合はおよそ280万組, テストデータ集合はおよそ10万組用意した. 学習データ集合の重複は10%ほど, テストデータ集合の重複は0.5%ほどだった.

[2] John's Chess Playground, https://tromp.github.io/chess/chess.html

5.1.2 手番情報の数値化

本研究では, ベース AI から得られる出力と手番情報を基に各候補手の特徴を抽出し CNN の入力とする. 以下に抽出した特徴とその表現方法を記述する. また各特徴はチェス盤のマス目に合わせて, それぞれ 64 個の数値を 8 行 8 列に並べて表現する.

特徴 1. 候補手の評価値

$\alpha - \beta$ pruning によって候補手に対応する最善応手系列の末端節点における評価値を得ることが出来る. これを各候補手の評価値として扱い, シグモイド関数によって区間 [1,-1] の単精度の浮動小数点数に変換している. 変換した数値を 64 個に複製し並べることで, 1 チャネルとする. また, 本研究で用いたサーバは同一節点の繰り返しが 3 度起こると引き分けとするよう設定されているため, 同一節点に繰り返し到達する場合, 評価値は末端節点における評価値ではなく, 0 が割り当てられる.

特徴 2. 候補手着手後の駒の配置

出力ログから得られる最善応手系列の最初の手を指した手番の駒の配置を 0, 1 で表現する. 駒の種類は相手プレイヤも合わせると 12 種類あるため, ポーン, ルーク, ビショップ, ナイト, クイーン, キングの位置を 1, それ以外を 0 で表す 12 チャネルと自プレイヤ, 相手プレイヤの全ての駒の位置を 1, それ以外を 0 とする 2 チャネルで表現する.

特徴 3. 候補手着手後のチェスの関する詳細なルール

- 候補手着手後のキャスリングの可否

  チェスではキャスリングと呼ばれる特殊な手を指すことが出来る. キャスリングを行うことが出来る条件は, キングとルークの間に駒が無い, キングの通り道に駒が利いていない, キングとルークを動かしていない, キングが攻撃されていないの 4 つである. プレイヤーは上記の条件を満たしているとき, キングを横に 2 マス動かし, ルークをキングを飛び越えた位置に動かすキャスリングと呼ばれる特殊な手を指すことができる. キャスリングはキング側で行うキャスリングとクイーン側で行うキャスリングの 2 つがある. 本実験では各候補手を着手した後に, プレイヤがその手番以降キャスリングが可能であるかどうかを表現するため, キングとルークを動かしていないかどうかのみに着目した.

  もし, クイーン側でキャスリング可能であるならばクイーン側のルークの位置に 1, そうでなければ 0, キング側でキャスリング可能であるならばキング側のルークの位置に 1, そうでなければ 0 とし, それ以外の位置はすべて 0 とする. このように両方のプレイヤのキャスリングの可否を 1 チャネルで表現する.

- 候補手着手後のアンパッサンの可否

  ポーンは基本, 前方 1 マス前に駒が無い場合, 前方 1 マス前に駒を進めることが出来るが, 2 マス先に相手の駒が無くゲーム開始時の位置から動いていないポーンに限っては, 2 マス前に進めることが出来る. 次の手番でその駒が通過したマスに相手プレイヤが駒を動かした場合, 前の手番に 2 マス動かされたポーンを取ることが出来る. ここではポーンが 2 マス前に動いた場合, ポーンが通過したマスに 1, それ以外を 0 とし, 1 チャネルでアンパッサンの可否を表現する. また, それ以外の場合は全ビットを 0 とした 1 チャネルによって表現する.

- 候補手着手後の 50 手ルールの適応

  ゲームが終盤に近づくと, お互い駒を取り合うため, 動かすことのできる駒が少なくなる. そのような状態でお互いが自分の駒が取られないように駒を動かすと勝負が拮抗し, 試合が冗長に伸びる. このような事態を避けるために 50 手ルールがある. これは両プレイヤが 50 手にわたってポーンを動かさず, 駒の取り合いが無い場合, 両者にゲームの勝敗をつける意思がないとみなし, 互いのプレイヤの申し出によって引き分けとするルールである. サーバは先手, 後手合わせて 100 手分, ポーンを動かさず駒の取り合いが無かった場合に引き分けとする. 本実験では, 着手 1 回あたり 0.01 とすることで, 100 手を区間 [0,1] の単精度の浮動小数点数に変換し, 64 個に複製し並べ 1 チャネルで表現する.

特徴 4. 候補手に対応した最善応手系列の末端の手番情報

候補手着手後から最善応手系列を指していったときの末端節点の特徴を表現する. 駒の配置やキャスリング, アンパッサンの可否, 50 手ルールも前述と同様に表現し, また最善応手系列の最後の手を指したときの手番が自プレイヤのとき全ビットが 0, 相手プレイヤのとき全ビットを 1 とした 1 チャネルで手番を表現する.

これらの入力の他に, 全ビット 1 とした 1 チャネルのビット列を加えた. なお, CNN が予測する結果は先手の勝ち, 引き分け, 負けであり, 候補手着手後が後手の手番ならば, 盤面, 評価値および最善応手系列を先手の手番と見做せるように変換した.

CNN に入力として与える特徴を限定することで, double-Fritz with boss に有用な特徴を分析したい. そこで表 1 のように異なる複数の入力を用意した.

入力はそれぞれ, A は候補手の評価値のみ, B は評価値と候補手着手後の駒の配置, C は評価値, 候補手着手後の駒の配置と候補手着手後のチェスの勝敗に関する詳細なルール, D は評価値, 候補手着手後の駒の配置と候補手着手後のチェスの勝敗に関する詳細なルールと候補手に対応した最善応手系列の末端節点に関する特徴とした. どの入力も 37 チャネルとして, 未使用のチャネルは 0 で埋めることとした.

表 1: 各入力

入力	特徴
A	1
B	1,2
C	1,2,3
D	1,2,3,4

5.1.3 CNN の構成

本研究では 3 つの異なる CNN(CNN1, CNN2, CNN3) を構築した. まず CNN1, 次に CNN2, 最後に CNN3 について述べる. 本研究で構築した CNN の構成を表 2 に示す. 表は各層におけるフィルタの次元, 出力されるユニットの次元, 出力数, パラメータ数, その層の出力で用いられている活性化関数を表している. なお表 2 で示している各層は, 表中の上下に位置する層と結合を持ち, 入力は上から下の層に伝播される. 入力層には採取した手番情報を数値化したものを与える.

CNN1 は畳み込み 9 層と全結合 1 層の計 10 層から構成される. 入力層から続く畳み込み層 1〜8 は C 枚のフィルタから成る. これらの畳み込み層は入力と出力のサイズが変わらないよう, ストライドは 1, パッディングは 1, フィルタのカーネルサイズは 3×3 とし, 活性化関数は ReLU を用いた. CNN の名称には括弧つきで C の値を記す. 畳み込み層 8 から続く 9 層目の畳み込み層のフィルタは 1 枚とした. この畳み込み層も同様にストライドは 1, パッディングは 1, フィルタのカーネルサイズは 3×3 である. 全結合層は選択された候補手の勝ち, 引き分け, 負けの確率分布に対応する 3 つの数値 p_1, p_2, p_3 を出力する. これらの数値は以下のソフトマックス関数によって z_1, z_2, z_3 に正規化される.

$$z_t = \frac{\exp(p_t)}{\sum_{i=1}^{3} \exp(p_i)} \tag{31}$$

CNN2 は次のような着想を元に構築する. 従来の double-Fritz with boss はボスである人間が 2 つの候補手を見て指し手を選択する. 一方, CNN1 は, 2 つの候補手についての特徴が同時に入力されるのではなく, 最善手, 次善手の特徴がそれぞれに入力されるモデルとなっている. そこで従来のものに倣って, 最善手, 次善手に関する特徴を一度に入力し, 最善手, 次善手を着手したときの勝ち, 引き分け, 負けを予測する CNN2 を構築する.

CNN2 には, 一度に 2 つの候補手に関する特徴を入力する. そのため, 入力のチャネル数は最善手に関する特徴 36 枚, 次善手に関する特徴 36 枚, 全ビット 1 としたチャネル 1 枚の計 73 枚となっている. CNN2 の全結合層のユニット数 P は 6 個で, 6 つの数値 $p_1, ..., p_6$ を出力する. この出力値からソフトマックス関数によって, それぞれ最善手の勝ち, 引き分け, 負けの確率分布 z_1, z_2, z_3 と次善手の勝ち, 引き分け, 負けの確率分布 z_4, z_5, z_6 を算出する. CNN2 のパラメータは, 入力したデータが最善手が選択された手番であった場合, 最善手の確率分布 z_1, z_2, z_3 から交差エントロピー誤差, 次善手が選択された手番の場合, 次善手の確率分布 z_4, z_5, z_6 から交差エントロピー誤差を算出して調整した.

CNN3 は CNN1 の全結合層の出力数を 1 つにし, 誤差関数を $K = 2$ の交差エントロピー誤差 (式 6 参照) としたものである. CNN1 との違いは出力される数値が結果でなく, 期待勝ち点 (勝ち 1, 引き分け 0.5, 負け 0) となっている点にある.

CNN1, CNN2 は最も大きい数値に対応する結果を予測とし, CNN3 は CNN3 は出力された数値が 0.66 より大きいとき勝ち, 0.33 より小さいとき負け, それ以外は引き分けを予測とする. どの CNN も予測と結果が一致していた場合, 予測が正しいものであったとしてカウントし, 1 バッチあたりの予測正答率を以下のように求める.

$$1\text{バッチ当たりの予測正答率} = \frac{\text{予測ラベルとラベルが一致した数}}{1\text{バッチあたりのデータ集合の組の数}}$$

本研究では予測正答率は, 1000 バッチ分の予測正答率の平均から算出した.

また, CNN1 の畳み込み 8 層のフィルタ数を変化させることでフィルタ数の増加に伴う CNN の性能の比較を行う. 性能の比較には, 入力を D, フィルタ数を 1,2,4,8,16,32 とした CNN 計 6 個を用いる.

これらの CNN のパラメータは, 深層学習用フレームワーク caffe(version 1.0.0) を用いて交差エントロピー誤差を最小化するように調整される. これは CNN1 と CNN2 は Adam によって, CNN3 は SGD によって行った.

5.1.4 Th 法

CNN を用いずに評価値とラベルの分布のみから勝ち, 引き分け, 負けの予測を試みた. まず, 2 つの閾値 a, b $(a > b)$ を定める. 評価値が a 以上のときは勝ち, 評価値が b 以下のときは負け, a

表 2: CNN の構造

層	フィルタの次元	ユニットの次元	出力数	パラメータ数	活性化関数
入力	-	8×8	I	0	-
畳込み 1	3×3	8×8	C	$9IC + C$	ReLU
畳込み 2	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 3	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 4	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 5	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 6	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 7	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 8	3×3	8×8	C	$9C^2 + C$	ReLU
畳込み 9	3×3	8×8	1	$10C$	ReLU
全結合	-	1×1	P	$65P$	softmax

より評価値が小さく, b より評価値が大きい時は引き分けと予測する. 学習データ集合のラベルの分布と評価値から, この予測の正答率が最も高くなるような, 閾値 a, b を定める. 学習データ集合によって定めた閾値を用いてテストデータ集合の勝ち, 引き分け, 負けの予測正答率を計測する. この手法を閾値 (Threshold) を用いていることから, Th 法と呼ぶ.

5.2 実験結果

各構成の CNN の学習曲線がおよそ収束したときの勝ち, 引き分け, 負けの予測の正答率を表 3 に示す. 不確かさは標準誤差から見積もられた 95%信頼区間である. 表 3 から CNN1-A(32) の予測正答率が, Th 法の予測正答率とほぼ同じとなったため, CNN1-A(32) は正しく評価値から勝ち, 引き分け, 負けを予測できていると考えられる. 評価値のみを入力とした CNN1-A(32) と比較して, 手番情報を入力に加えた CNN1-B(32), CNN1-C(32), CNN1-D(32) が良い結果を示した. また, CNN1-B(32) と比較して, 詳細なチェスのルールを追加した CNN1-C(32) や, さらに最善応手系列の末端節点における駒の配置や詳細なチェスのルールを追加した CNN1-D(32) の予測正答率には有意な差が確認することはできなかった.

また, CNN1-D(1), CNN1-D(2), CNN1-D(4) の予測正答率は有意な差は確認できないが, CNN1-D(4) と CNN1-D(8), CNN1-D(16), CNN1-D(32) との比較から畳込み層のフィルタ数の増加に従って徐々に予測正答率が増加していることが伺える. また, Th 法と CNN1-D(1) の予測正答率の比較から, 少なくとも小さい規模の CNN でも評価値の閾値以上の特徴をが認識できていると推測される. 他に CNN2-D(32) は, CNN1-D(32) と異なり, 一度に最善手と次善手の確率分布を出力し選択された候補手の勝ち, 引き分け, 負けの予測を行っているが, 予測正答率という観点においては CNN1-D(32) との有意な差を確認することはできなかった.

ここで Th 法と比較して, CNN1-D(32) の予測正答率が有意に高いことから, 駒の配置や詳細な特徴, 末端節点の特徴などを入力に追加することによって, CNN がどのような予測を行っているかを調査する. それぞれの手法による勝ち, 引き分け, 負けの予測と正答の比較を表 4, 表 5 と表 6 に示す.

表 3: 各手法による勝ち, 引き分け, 負けの予測正答率

手法	予測正答率
Th 法	0.867 ± 0.002
CNN1-A(32)	0.867 ± 0.002
CNN1-B(32)	0.876 ± 0.002
CNN1-C(32)	0.878 ± 0.002
CNN1-D(1)	0.872 ± 0.002
CNN1-D(2)	0.872 ± 0.002
CNN1-D(4)	0.872 ± 0.002
CNN1-D(8)	0.876 ± 0.002
CNN1-D(16)	0.877 ± 0.002
CNN1-D(32)	0.878 ± 0.002
CNN2-D(32)	0.877 ± 0.002
CNN3-D(32)	0.874 ± 0.002

表 4: CNN1-D(32) の予測と正答の比較

	勝ち (正答)	引き分け (正答)	負け (正答)
勝ち (予測)	18,242	1,689	231
引き分け (予測)	4,177	48,091	3,869
負け (予測)	384	1,940	21,177

表 5: Th 法の予測と正答の比較

	勝ち (正答)	引き分け (正答)	負け (正答)
勝ち (予測)	17,455	1,446	82
引き分け (予測)	5,094	48,346	4,489
負け (予測)	254	1,928	20,706

表 6: Th 法の予測と CNN1-D(32) の予測の比較

	勝ち (CNN1-D(32))	引き分け (CNN1-D(32))	負け (CNN1-D(32))
勝ち (Th 法)	18,405	578	0
引き分け (Th 法)	1,757	54,798	1,374
負け (Th 法)	0	761	22,127

表 4 と表 5 を比較すると, Th 法よりも CNN1-D(32) の勝ち, 負けの予測数が増え, 引き分けの予測数が減っている. また表 6 より, Th 法と CNN1-D(32) の予測を比較すると, Th 法が勝ちと予測しているときに CNN1-D(32) が負け, Th 法が負けと予測しているときに CNN1-D(32) が勝ちと予測することは無いことが伺える. これより, Th 法で行われている引き分けの予測が CNN1-D(32) の勝ち, 負けの予測に流れていることが伺える.

6 CNNによって構築したボスとベースAIによるdouble-Fritz with bossの対戦実験

6章では5章で構築したCNNを用いて指し手の選択を行うボスを構築し, その性能を調査する.

6.1 提案手法

Double-Fritz with bossにおけるボスのAIを構築するにあたって, 重要となるのはなにをもってボスに最善手, 次善手の選択を行わせるかである. 3章で述べたように先行研究では指し手の選択を行うとき, 主に各候補手に対する投票数や評価値などに焦点を当てていた. しかし, double-Fritz with bossのボスは1つのベースAIから得た複数の候補手から指し手を選択する. そこで, 候補手の評価値とチェスの戦略において有効となりうるような特徴をCNNに入力し, 出力された勝ち, 引き分け, 負けの予測結果からボスに指し手の選択を行わせたい. 前章ではStockfish7が出力する評価値以外に, 候補手着手後の駒の配置, 現手番までの指し手の系列などから抽出できるチェスのルールに関する特徴や末端節点における特徴などを入力として勝ち, 引き分け, 負けを予測するCNNを構築した. このCNNの出力から得られる勝ち, 引き分け, 負けの確率分布から候補手の期待勝ち点を算出する. 最善手, 次善手の期待勝ち点を算出し, 期待勝ち点が高い方の手を選択するボスを構築し, 最善手のみを選択するボスと対戦を行わせる. 対戦結果から得られる期待勝ち点, ボスに選択された最善手, 次善手に関する解析やdouble-Fritz with bossの構築に最適なCNNの構成などについての定量的な分析を行う.

6.2 期待勝ち点

構築したボスの性能は予測正答率とは別に期待勝ち点によって見積もることが出来る. チェスの試合の結果は勝ち, 引き分け, 負けの3種類である. 引き分けとなる条件は同一節点の繰り返し, 50手ルールによる引き分けやステイルメイトなどがある. 勝ち点は勝ちが1, 引き分けが0.5, 負けが0として扱われ, その期待値を取ることにより性能を数値として算出できる. これを期待勝ち点と呼ぶ. 試合の序盤はopening-bookから指し手を選択するため, 試合はそれぞれ異なり独立していると考えることができる. ここでopening-bookのランダム性による試合の結果の偏りを期待勝ち点の誤差として考慮する. 試合数は3000試合として, 先手の試合を1500試合, 後手の試合を1500試合の同数とすることで手番による勝ち点への影響は考えないものとする. ここで, 試合の結果を母集団とした正規分布と仮定し, 無作為に3000個の結果を選出したときの勝ち点の誤差を計測する. 勝ちの試合数を N_w, 引き分けの試合数を N_d, 負けの試合数を N_l とすると勝ちの確率 P_w, 引き分けの確率 P_e, 負けの確率 P_l, 推定値である勝ち点の平均 μ, 分散 σ^2,

標準誤差 E は以下の数式で求まる.

$$P_w = \frac{N_w}{N_w + N_d + N_l}$$
$$P_d = \frac{N_d}{N_w + N_d + N_l}$$
$$P_l = \frac{N_l}{N_w + N_d + N_l}$$
$$\mu = P_w + \frac{P_d}{2}$$
$$\sigma^2 = P_w(1-\mu)^2 + P_d(\frac{1}{2} - \mu)^2 + P_L(0-\mu)^2$$
$$E = \sqrt{\frac{\sigma^2}{3000}}$$

よって勝ち点の推測値の 95%信頼区間は, $\mu \pm 1.96 \times E$ で見積もることができる.

6.3 実験結果

各 CNN を用いたボスを構築し, 最善手のみ選択を行うボスと 3000 試合の対戦を行った結果が表 7 である. 表 7 より, CNN1-A(32), CNN1-B(32), CNN1-C(32), CNN1-D(32), CNN3-D(32)

表 7: 各 CNN によって構築したボスの期待勝ち点

CNN	期待勝ち点
1-A(32)	0.50 ± 0.02
1-B(32)	0.48 ± 0.02
1-C(32)	0.47 ± 0.02
1-D(1)	0.49 ± 0.02
1-D(2)	0.47 ± 0.02
1-D(4)	0.47 ± 0.02
1-D(8)	0.48 ± 0.02
1-D(16)	0.49 ± 0.02
1-D(32)	0.49 ± 0.02
2-D(32)	0.34 ± 0.02
3-D(32)	0.48 ± 0.02

の期待勝ち点に有意な差がないことが伺える. また CNN1-D(32) の規模を小さくした構成の CNN1-D(1), CNN1-D(2), CNN1-D(4), CNN1-D(8), CNN1-D(16) と CNN1-D(32) の期待勝ち点についても有意な差がないことが伺える. 表 3 と比較すると各候補手の勝ち, 引き分け, 負けの予測精度のわずかな向上が必ずしも期待勝ち点の向上に結び付いている訳ではないことがわかる. また, 予測精度と期待勝ち点の相関をより詳しく調査するため, バッチ処理回数ごとの CNN1-D(32) の期待勝ち点と予測正答率についての調査結果を示したものが図 6 である. 図 6 の横軸はバッチ処理回数, 縦軸は期待勝ち点を表していて, 赤い点が期待勝ち点, 点に付随する赤い線が期待勝ち点の 95%信頼区間における標準誤差, 緑の点がテストデータ全体の予測正答率である. バッチ処理回数 10^5 から 10^6 で急激に増加していることが見受けられる. また, それに伴い期待勝ち点も増加していることが確認できる. このことから, 予測正答率と期待勝ち点に

はある程度の相関があることが推測できる. しかし 10^7 以降は, 期待勝ち点はバッチ処理回数に伴いわずかに増加しつづけているが, 期待勝ち点の変動に有意な差は確認できないため, これ以上のバッチ処理回数の増加に伴う予測正答率と期待勝ち点の大幅な増加は見込めないと考えられる.

また表 3, 表 7 より, 畳込み層のフィルタ数を 1 から 32 まで増加させても, 予測精度に大きな向上は確認できず, 期待勝ち点にも有意な差が確認できなかった. 以上の結果と, 3 章で述べたチェス AI である AlphaZero が, 畳み込み層のフィルタ数は 256, かつ層の数が 20 以上であるようなより深い構造の CNN が用いていることから, チェスの特徴を CNN で十分に表現し ボスの性能を向上させるためにはより大規模な CNN を構築する必要があるといえる.

他に,CNN2-D(32) を用いたボスの期待勝ち点は他のボスと比べて有意に期待勝ち点が低い結果となった. CNN2-D(32) を用いたボスによる次善手選択確率を調べたところ, ボスが候補手の選択を行う手番のうち, 43%が次善手を選択した手番であった. しかし, なぜ CNN2-D(32) を用いて構築したボスが他のボスと比較して次善手選択確率が高く, 有意に性能が低いのかを明らかにすることは出来なかった.

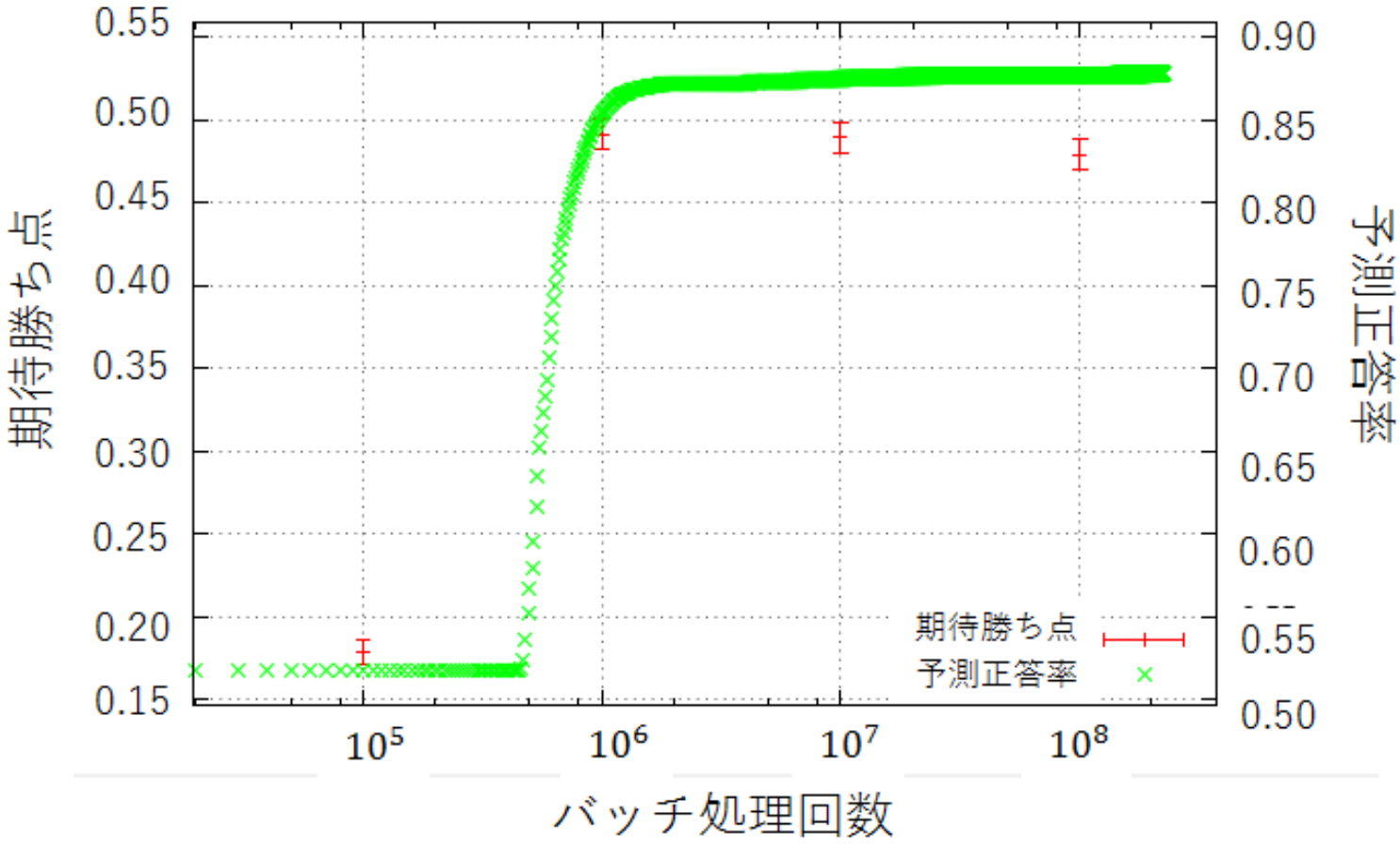

図 6: バッチ処理回数における予測正答率と期待勝ち点の推移

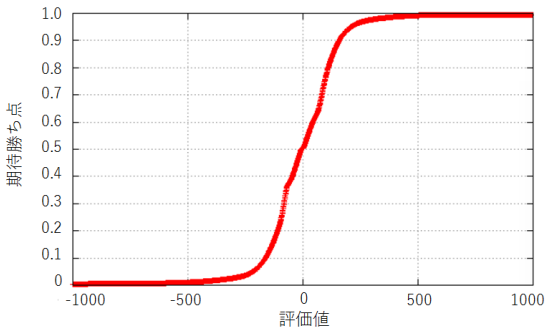

図 7: CNN1-A(32) に入力した評価値に対応する期待勝ち点

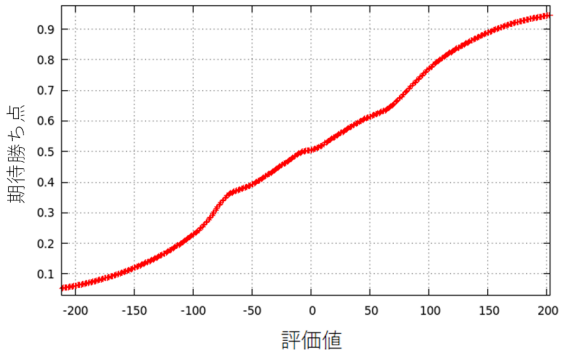

図 8: 図 7 を拡大したもの

ここで CNN1-A(32) の性能を解析するため, 評価値を模した区間 [-1000, 1000] までの整数値を正規化し CNN1-A(32) に入力して, 期待勝ち点を算出した. その結果が図 7, 図 8 である. 横軸は CNN1-A(32) に入力した評価値, 縦軸は入力して得られた出力から算出された期待勝ち点である. 図 8 から, およそ入力した評価値 -150 で期待勝ち点が 0.1, 評価値 150 ほどで期待勝ち点が 0.9, 評価値 -50 で期待勝ち点が 0.4, 評価値 50 で期待勝ち点が 0.6 となっていることがわかる. また図 8 から評価値が 0 のとき, 期待勝ち点が引き分けを表す 0.50 となっていることが伺える. このことから, CNN1-A(32) は実際に評価値から試合の結果を予測し, 期待勝ち点を算出できていることがわかる. また, 本実験の 1 つのプレイから学習データを採取する条件の 1 つである, 最善手, 次善手の評価値が区間 [-1000, 1000] であることは, 学習データ採取の条件として適切ではなかったと考えられる. 学習データ採取の条件として, 最善手と次善手の評価値がより狭い区間にあるような手番を採取すると, NN の学習はより勝敗の趨勢に影響するような候補手のプレイ結果予測に特化したものとなる可能性がある.

また, CNN3-D(32) の予測の様子を調査した. CNN3-D(32) にテストデータを入力したときの出力値の分布を勝ち, 引き分け, 負けの試合でそれぞれに示したものが図 9 である. 図 9 の横軸は, CNN3-D(32) の出力値, 縦軸は出力値の分布の割合となっている. 勝ちの試合の出力値の分布を赤い棒, 引き分けの試合における出力値の分布を緑の棒, 負けの試合における出力値の分布を青の棒で示している. 出力値の割合は数値 0.01 毎に次式のように求めた. 次式では勝ちの試合における出力値の割合を例として挙げている.

$$= \frac{\text{勝ちの試合で CNN3-D(32) の出力値が } x \sim x + 0.009 \text{ となった数の割合}}{\text{勝ちの試合で CNN3-D(32) の出力値が } x \sim x + 0.009 \text{ となった数の合計}}{\text{勝ちの試合の手番の合計}}$$

ここで CNN3-D(32) で算出される期待勝ち点は他の CNN と異なり, 全結合層の出力値であることに注意する. 図 9 のように勝ちのときの手番の期待勝ち点は, およそ 1.0, 引き分けの手番のときの期待勝ち点はおよそ 0.5, 負けの手番のときの期待勝ち点は, およそ 0 に近い値を出力していることが多いことが読み取れる. しかし, 勝ち, 引き分け, 負けのどの試合においても, いくらか 0.5 に近い数値を出力していることがあり, このことから序盤に多いような, 評価値が 0 に近いいくつかのデータにおいては, 勝ち, 引き分け, 負けを予測することが困難であることが確認できる.

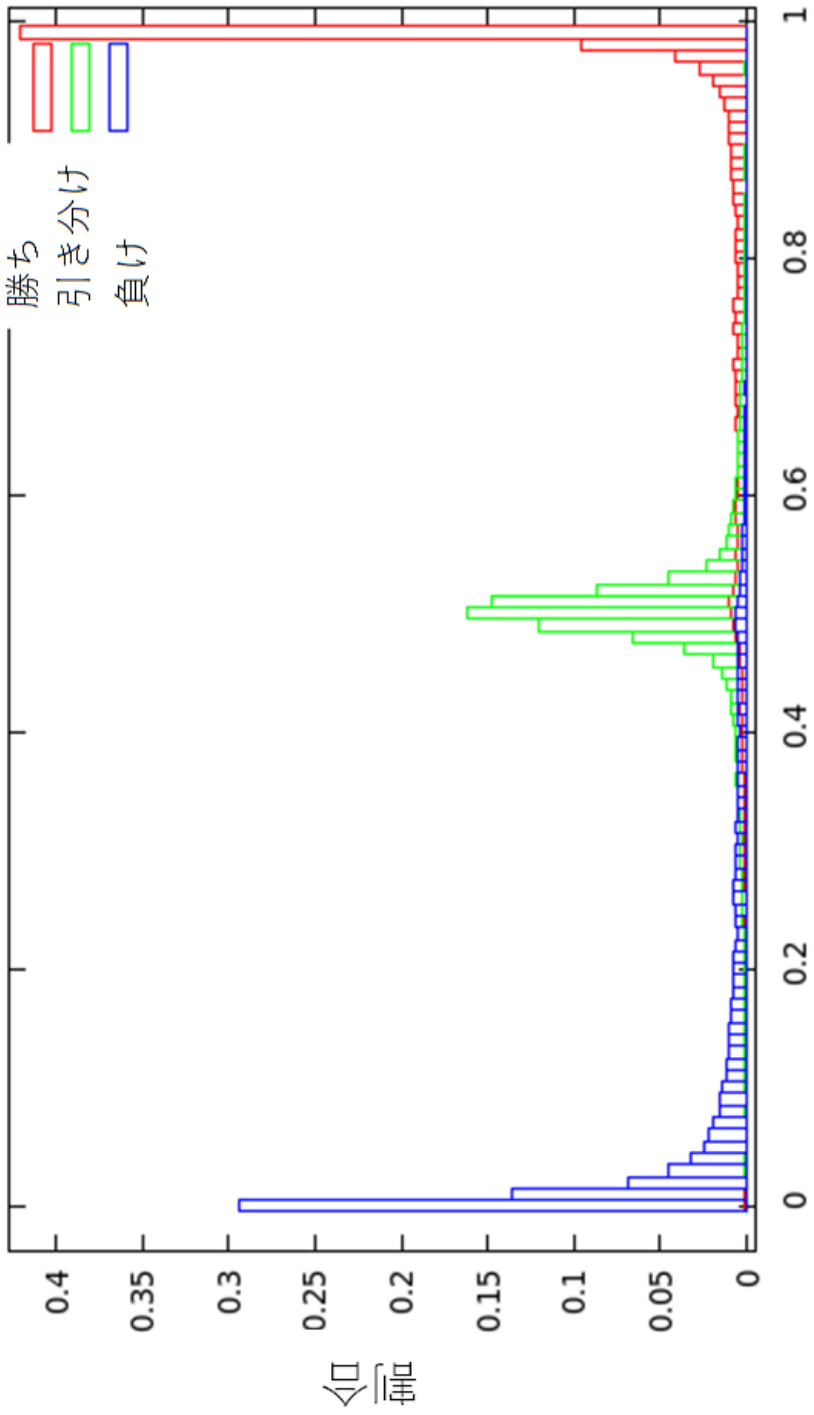

図 9: CNN3-D(32) の出力の分布をラベル別に示したもの

本実験で構築したボスを詳しく分析するために, CNN1-D(32) をボスとして用いた AI と常に最善手を選択するボスを用いた AI によるゲームのプレイ 3000 回からデータの解析を行った.

CNN1-D(32) を用いて着手を行う手番がどの程度存在するのかを調査するため, opening-book を用いた手番の数, CNN1-D(32) が選択を行う, 最善手と次善手の評価値が区間 [-1000, 1000] におさまる手番の数について調査した結果を表 8 に示す. 表 8 から実際に, 全体の 60%以上の手番においてボスが最善手と次善手の 2 つの候補手を考慮して指し手を選択していることがわかる. またボスによって指し手が選択されている手番のうち, およそ全体の 17%程度の手番で, 実際に次善手が選択されていることが確認されている. これらのことから, CNN1-D(32) による候補手の選択は期待勝ち点の変動に対して十分寄与していると考えられる.

表 8: 3000 回のプレイにおける CNN1-D(32) の着手選択方法の分布 (総着手数は 264,494)

着手の分類	数
opening-book	59,542
評価値区間内の着手	178,342
次善手の着手	31,383

他には, CNN1-D(32) による次善手の選択がどの程度されているかを調査するため勝ち, 引き分け, 負けの試合で次善手が選択されている確率について調査した. その結果を図 10 に示す. 赤い点が勝ちの試合で次善手が選択された確率, 緑の点が引き分けの試合で次善手が選択された確率, 青い点が負けの試合で次善手が選択された確率となっている. 点に付随している線は 95%信頼区間を表す標準誤差となっている. また, 次善手の選択確率は 10 手毎に以下のように求めた.

$$\text{手数 } x\text{〜}x+9 \text{ までの次善手の選択確率} = \frac{\text{手数 } x\text{〜}x+9 \text{ の手番で選択された次善手の数}}{\text{手数 } x\text{〜}x+9 \text{ の手番の数}}$$

式で示している手番の数はボスに選択された手番のことを表している. チェスにおける手番は通常, 先手後手あわせて 1 手と数えるが, 本研究では先手, 後手の手番をあわせて 2 手と別々にカウントしている. どの結果の試合もおよそ手数が 20〜60 ほどの手番ではおよそ 1 割近く次善手が選択されていることがわかる. 勝ち, 負けの試合においては手数に伴いゆるやかに次善手選択確率が増加している. 引き分けの試合においては手数に伴う次善手選択確率の増加が顕著に表れていて, 手数 140 を超えるとおよそ 4 割ほどの確率で次善手が選択されている. これは手数が増えるにつれて駒の数が少なくなる傾向があり, 最善手と次善手のどちらを打ってもプレイ結果に与える影響が小さいためと考えられる.

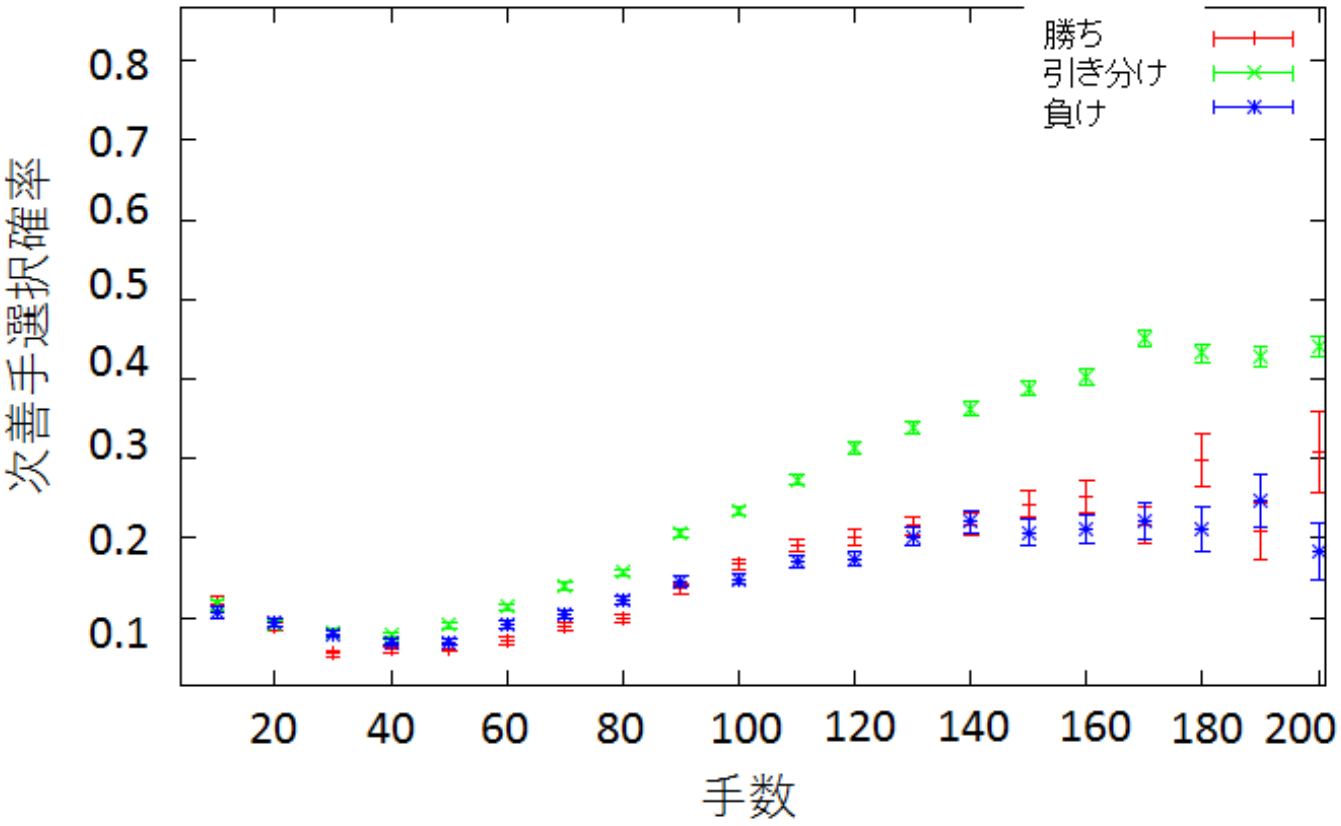

図 10: 各手数における次善手選択確率の推移

ここで, 次善手の選択がどの程度期待勝ち点を低下させ得るのかを確かめるために, 候補手の評価値の差が一定の数値 v 以下のとき無作為に選択するようなボスを用いて対戦実験を行った.v を 1,2,4,8,16,32 として常に最善手を選択するボスと 3,000 戦させた結果を表 9 に示す. v が 4 の場合は期待勝ち点に 0.5 と有意な差がなく, かつ次善手選択確率も CNN1-D(32) と同程度であった. 従って, 本研究で構成した CNN を用いたボスの性能は, 評価値を見て次善手を時折選択するような方法と同程度であったといえる.

表 9: 候補手の評価値の差が v 以下のとき, 無作為に指し手を選択するボス AI の期待勝ち点と次善手選択確率

v	次善手選択確率	期待勝ち点
1	0.13	0.49±0.02
2	0.15	0.48±0.02
4	0.17	0.48±0.02
8	0.23	0.48±0.02
16	0.30	0.48±0.02
32	0.37	0.48±0.02

実際にどのような場面で次善手が選択されているかをより詳しく調査するため勝ち, 引き分け, 負けの試合において最善手, 次善手を選択した手番の評価値について調査した結果を図 11,12 に示す.

図 11, 図 12 の横軸は各手番における最善手の評価値, 縦軸は最善手と次善手の評価値の差となっている. 各点はゲーム 3000 プレイでボスに選択された手番のうち, それぞれ勝ちのものを 100 点, 引き分けのものを 100 点, 負けのものを 100 点ずつ無作為に採取して描かれた. 図 11 の赤い点は CNN1-D(32) が勝ち, かつ最善手を選択した手番の評価値の分布, 緑の点は引き分け, かつ最善手を選択した手番の評価値の分布, 青い点は負け, かつ最善手を選択した手番の評価値の分布となっている. 同様に図 12 の赤い点は CNN1-D(32) が勝ち, かつ次善手を選択した手番の評価値の分布, 緑の点は引き分け, かつ次善手を選択した手番の評価値の分布, 青い点は負け, かつ次善手を選択した手番の評価値の分布となっている. 図 11, 図 12 から実際に次善手が選択されているのは最善手と次善手の評価値の差が 0, もしくは僅かな差であるときで, さらに最善手の評価値が 0 の時により顕著に表れていた.

一方, 最善手の評価値が大きい, または小さい時に次善手を選択されている手番については, 比較的評価値の差が大きい場合でも次善手が選択されている. これは評価値が大きい, または小さい時はおよそゲームの趨勢が決しており, 次善手の選択によるゲームの結果の変動が起こらないとボスが判断しているためだと考えられる.

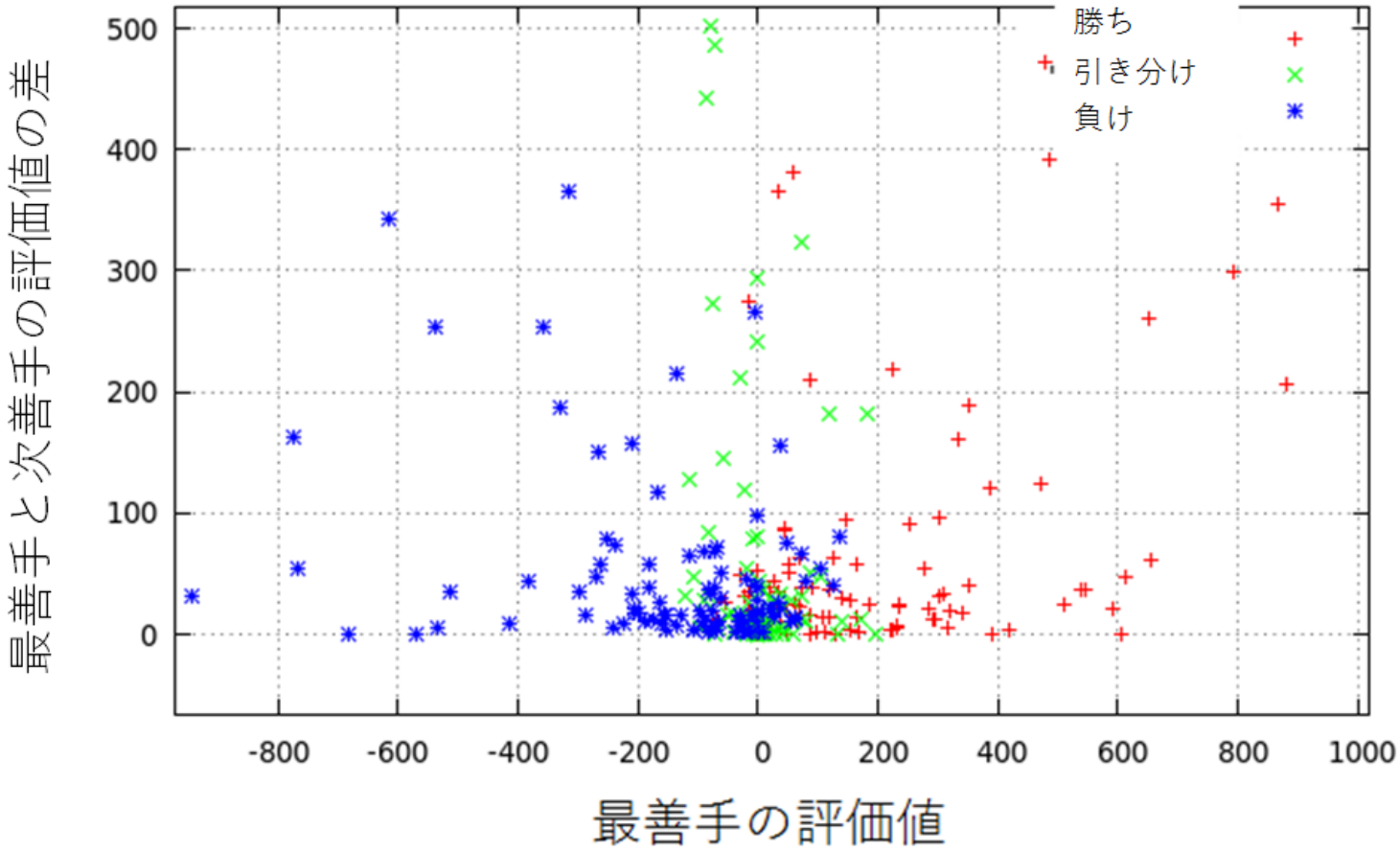

図 11: CNN1-D(32) が勝ち, 引き分け, 負けでかつ最善手を選択した手番の最善手, 次善手の評価値の分布

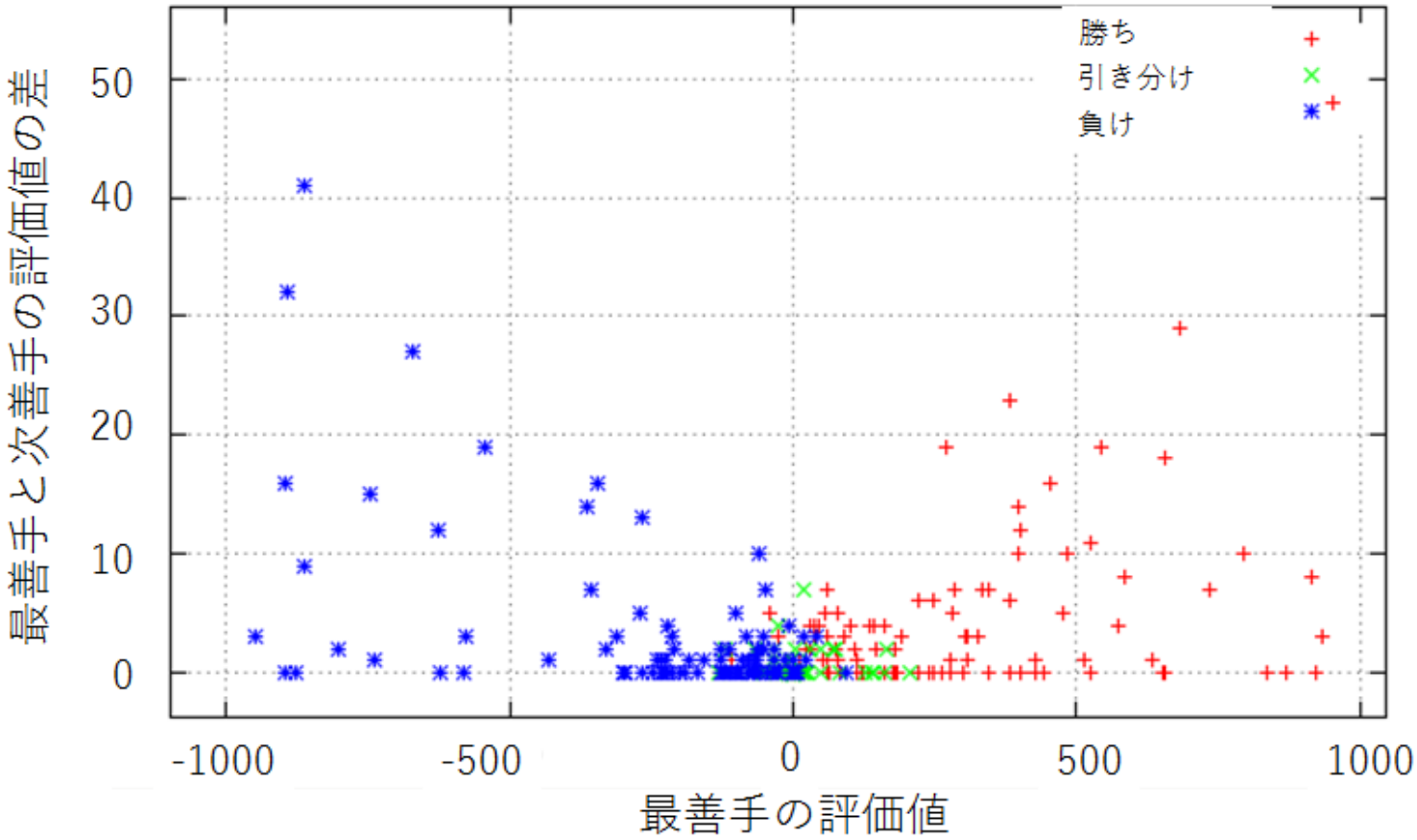

図 12: CNN1-D(32) が勝ち, 引き分け, 負けでかつ次善手を選択した手番の最善手, 次善手の評価値の分布

7　おわりに

本研究では, CNN を用いてチェスのプレイの結果である勝ち, 引き分け, 負けの予測を行った. 構築した CNN はベース AI の出力する評価値に加えて手番情報を入力する. この CNN は評価値と対戦結果の分布のみから結果を予測する単純な手法以上の予測精度を持つことが確認された. また CNN の大規模化に伴い, 予測精度は有意に向上した.

そして CNN の出力から期待勝ち点を推定し, 指し手を選択する double-Fritz with boss のボスを構築した. ボスの予測正答率と期待勝ち点の変動にはある程度の相関がみられたが, どの構成の CNN を用いても, 期待勝ち点は 0.5 と有意な差がなく, ベース AI よりも強い AI を生成するには至らなかった. また, CNN を用いて構築したボスの性能は, 評価値をみてときおり次善手を選択するような方法と同程度だった.

また, サイズを変化させた CNN の期待勝ち点に大きな変化はなく, フィルタ数 32 の畳み込み層 8 層の CNN では, 手番情報からチェスの優劣判断に有効な特徴を抽出するのは難しいことがわかった. しかし, フィルタ数の増加に伴い予測正答率が増加したことから, より大規模な CNN を構築することににによって予測正答率を大きく向上させることができれば, 期待勝ち点を向上させ得るのではないかと考えられる.

これらの結果をうけて, より良い候補手選択を行うためには, さらなる大規模な CNN の構築や, ラベルの変更, 学習データ点の採取方法の見直しなどを行う必要がある.

8　謝辞

本研究に際して, 研究指導や修士論文の推敲などの様々なご指導を頂きました保木先生に深く感謝致します. また, 多くのことをご指導頂きました村松先生や高橋先生, ゲームゼミの研究室の皆様に感謝致します.

参考文献

[1] 赤間世紀. 集合知入門. I/O books. 工学社, 2014.

[2] Ingo Althöfer and Raymond G. Snatzke. Playing games with multiple choice systems. In *International Conference on Computers and Games*, pp. 142–153. Springer, 2002.

[3] Kristian T. Spoerer, Toshihisa Okaneya, Kokolo Ikeda, and Hiroyuki Iida. Further investigations of 3-member simple majority voting for chess. In *International Conference on Computers and Games*, pp. 199–207. Springer, 2013.

[4] Kunihito Hoki, Seiya Omori, and Takeshi Ito. Analysis of performance of consultation methods in computer chess. *Journal of Information Science and Engineering*, Vol. 30, No. 3, pp. 701–712, 2014.

[5] 大森誠也, 保木邦仁, 伊藤毅志. チェスプログラムを用いた合議アルゴリズムの効果の検証. 研究報告ゲーム情報学 (GI), Vol. 26, No. 5, pp. 1–7, 2011.

[6] 小幡拓弥, 塙雅織, 伊藤毅志. 思考ゲームによる合議アルゴリズム～単純多数決の有効性について～. 研究報告ゲーム情報学 (GI), Vol. 22, No. 2, pp. 1–5, 2009.

[7] 伊藤毅志, 小幡拓弥, 杉山卓弥, 保木邦仁. 将棋における合議アルゴリズム—多数決による手の選択. 情報処理学会論文誌, Vol. 52, No. 11, pp. 3030–3037, 2011.

[8] 杉山卓弥, 小幡拓弥, 斎藤博昭, 保木邦仁, 伊藤毅志. 将棋における合議アルゴリズム—局面評価値に基づいた指し手の選択. 情報処理学会論文誌, Vol. 51, No. 11, pp. 2048–2054, 2010.

[9] Danilo S. Carvalho, Minh L. Nguyen, and Hiroyuki Iida. An analysis of majority voting in homogeneous groups for checkers: Understanding group performance through unbalance. In *Advances in Computer Games*, pp. 213–223. Springer, 2017.

[10] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, and Thore Graepel. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[11] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, Vol. 356, No. 6337, pp. 508–513, 2017.

[12] Gerald Tesauro. TD-gammon: A self-teaching backgammon program. In *Applications of Neural Networks*, pp. 267–285. Springer, 1995.

[13] 岡谷貴之. 深層学習. MLP 機械学習プロフェッショナルシリーズ. 講談社, 2015.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A　UCI プロコトルのコマンド

ここでは, 本研究で用いた UCI プロトコルの通信コマンドについて述べる.

サーバからチェス AI への通信文字列

ucinewgame

チェス AI に対して新規の対局が始まったことを知らせるコマンドである.

uci

チェス AI に対して探索に関するパラメータの開示を求めるコマンドである. 相手の手番で探索を行う ponder の使用の有無, チェス AI が探索の効率化に用いるハッシュテーブルのサイズ, 探索に用いる並列スレッド数,MultiPV 機能などの初期値, 変更可能なパラメータの最小値, 最大値などが表示される.

setoption name (文字列 1) value (文字列 2)

チェス AI の探索に関するパラメータの変更を求めるコマンドである. (文字列 1) には変更するパラメータの名前, (文字列 2) には変更後のパラメータの値が入る.

go

チェス AI に探索を行わせるコマンドである. 末尾に特定の文字列を加えることで、探索節点数や探索時間, 探索の深さなどに制限を加えることができる. このコマンドの後, サーバはチェス AI から指し手を示す文字列である bestmove を含む文字列を受け取るか, 探索時間に達するまで待機する.

チェス AI からサーバへの通信文字列

bestmove (文字列 1)

サーバに指し手を返すコマンドである.　(文字列 1) にはチェスの指し手が入る. チェスの指し手を示す表記法はいくつか存在するが、UCI プロコトルでは long algebraic notation が採用されている. この表記法では, 指し手は 4 文字, もしくは 5 文字で表現され, 手を指す前の指し手の配置と指した後の指し手の配置, またはポーンが昇格する駒の種類を表す文字列の連結によって表現される.